TITLE:

数式のインテリジェントな線形入力方式 (数学ソフトウェアと教育 : 数学ソフトウェアの効果的利用に関する研究)

AUTHOR(S):

福井, 哲夫

CITATION:

福井, 哲夫. 数式のインテリジェントな線形入力方式 (数学ソフトウェアと教育 : 数学ソフトウェアの効果的利用に関する研究). 数理解析研究所講究録 2012, 1780: 160-171

ISSUE DATE:

2012-03

URL:

http://hdl.handle.net/2433/171821

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
第1780巻 2012年 160-171

数式のインテリジェントな線形入力方式

武庫川女子大学　福井哲夫（Fukui Tetsuo）
Mukogawa Women's University

1　はじめに

1980年代より教育の情報化が推進され，2011年4月には，文部科学省より「教育の情報化ビジョン」が打ち出され[1]，中学・高等学校においてデジタル教科書が導入されようとしている．教師用のデジタル教科書は一斉授業において電子黒板で提示したり，教師があらかじめ準備した電子教材と組み合わせて分かり易く教示するためにさまざまなコンテンツが開発され，大変便利になりつつある．また，ビジョンによれば，学習者用デジタル教科書には従来の教育内容をデジタル化するだけでなく，電子閲覧機器においてノートやメモをとる機能や練習問題等の個別学習機能が求められている．

しかし，理数系教育において問題となるのが数式のデジタル入出力である．特に，電子的個別学習の場面では，生徒・学生も数式を扱う必要性がますます高まりつつある[2],[3]．数式は文と異なり，2次元的な表記構造をもっており，従来のデジタル端末は数式の表記を十分考慮されておらず，数式の取り扱いを困難にしている．

一方，数式処理システム(CAS)を理数系教育に利用しようとする取り組みも多く報告されるようになってきた．しかし，生徒に直接CASを使わせて，練習問題を解かせるやり方には良くない面もある．なぜなら，結局そこで生徒が学んだことは，本来学ぶべき数学的概念ではなく，与えられた問題をCASの文法に従った命令に翻訳する技術とSolveやPlotコマンドなどを入力すればCASが答えを出してくれるという事実だけである．

私が理想とする形態はまず，第1に，パソコンやデジタル端末を紙と鉛筆の代わりに，計算ノートを記録し，生徒自身が答えを導く思考の道具[4]となるようにすること．第2に，生徒の学習活動の途中計算や答えに対して，その理解度を判定し助言を与える処理のためにCASを活用することである．

そのためには，まず，数式のデジタル入力を平易にする必要がある．本研究は，それを解決するための新しい数式入力方式を提案することが目的である．

第2章では，従来の技術とその問題点について紹介し，第3章では，数式のインテリジェントな入力方式を提案する．第4章で，提案方式の望ましい実施例を示し，最後に，本研究の将来展望をまとめる．

2　従来の技術

ここでは，数式入力のための従来の技術（現状）について紹介し，その問題点を述べる．

2.1 用語

従来技術を説明するためにまず，いくつかの用語を定義しておく．

「線形文字列形式(linear string format)」という言葉は，例えばポーランド式プレフィックス形式（ポーランド記法）や逆ポーランド記法，TeXやLaTeXなど，線形表記法を用いた数式の線形テキストに基づく表現を指す．ポーランド式プレフィックス形式は，関数開始文字を含む形式であり，関数開始文字の後に，例えば，分子，分離文字，分母，および関数終了区切り文字が続く．LaTeXによる線形文字列形式の式の例は，

￥frac{1}{a ^ 2+1} （aの二乗足す1分の1）

である．

「2次元形式」という言葉は，数式が，非線形表記法を用いて表される形式を指す．例えば，分子，分離文字，分母を上下に，指数部を上付き添え字など2次元的に配置する．2次元形式の例は，

$\frac{1}{a^2+1}$ （aの二乗足す1分の1）

である．

この2次元形式で表された数式は「数式最小単位記号」自身であるか，「演算子」およびその演算子が作用している1個または複数のオペランド（引数）によって構成されている．オペランドはまた，一つの数式である．

「数式最小単位記号」と言う言葉は，数，変数，文字などを指す．

「演算子」と言う言葉は，本論文では，2次元形式の構造をきめる関数，二項演算子，積分演算子，縮約演算子，括弧，冪乗，添え字などを指し，演算子記号および／またはオペランド表記との相対的配置および大きさによってその構造が決まる．

以下では，2次元形式で表された数式を構成している数式最小単位記号または演算子を「数式要素」と呼ぶ．

2.2 数式入力の現状と問題点

現在用いられている数式入力の方法としては，次の3つのタイプが代表的である．

従来タイプ1：線形文字列コンパイル方式

これは数式表示の指示を線形文字列形式で入力するタイプである．

[具体例] TeX, LaTeX[5], MathML[6] など

[問題点] タイプ1の方法は，通常，コンパイルによるコンピュータ内部の処理が行われないと数式として表示できない上に，数式整形指示コマンドの文法を覚えなければ入力できないため利用者に負担がかかり，それをテキスト入力する作業が繁雑である．

例えば，LaTexでは数式 $\frac{\sin x^2}{2}$ を表すために￥frac{￥sin x ^ 2}{2} と入力する．

従来タイプ2：GUIテンプレート方式

ユーザが，2次元形式の式を作るために様々な数式パターンのツールバーアイコンの中から選択を行うことを要求するWYSIWYGエディタなどのタイプである．

[具体例] 文書処理アプリケーションに付属する数式エディタ [7],[8] など

[問題点] タイプ2の方法は，常に数式表示を確認しながら数式を入力できるため優れているが，人の自然な入力手順と異なり，先にユーザが数式構造を把握し，対応する配置パターンをGUIテンプレートから選択入力しなければならないため，複雑で長い数式に不慣れなユーザにとって煩わしい場合がある．例えば，分数 $\frac{1}{2}$ を入力したい場合，まずテンプレートから分数構造 $\frac{\square}{\square}$ を選択し，次にカーソルを分子の位置に移動して数1を入力，カーソルを分母に移動し数2を入力，の順に行う．

$$\frac{\square}{\square} \rightarrow \frac{1}{\square} \rightarrow \frac{1}{2}$$

従来タイプ3：手書き入力方式

ペン入力デバイスまたはタッチパネルを用いて，手書きで入力するものである．

[具体例] InftyEditor[9],[10], Windows7[11] の数式入力パネルなど

[問題点] タイプ3は，数式の最も自然な入力方式であるが，特別な入力デバイス（ペンタブレット等）が必要となる．また，数式入力作業途中でショートカットを導入することは難しく，入力効率が上がらない上に，テキスト形式の入力に比べて，間違えた際の修正が面倒である．また，通常の文章はキーボード入力の方が効率的なため，文章と数式が混在した入力の場合に，手書きとキー入力の切替操作が面倒になる．

3 数式のインテリジェントな入力方式の提案

本研究は第2章で述べた従来技術の問題点を解決し，ユーザによる数式の入力に関する操作性をより向上させるために新しい数式入力方式を提案する．

3.1 線形文字列表記法（新提案）

本研究で提案する線形文字列形式表記法を，以下の（1），（2）で定める．

(1) 所望する数式を構成している各数式要素を然るべきキー文字／文字列（例えば表1）で表し，

(2) 所望する数式の構造を決定している演算子に応じた然るべき順番（例えば表2）に，区切りなく線形に並べる．

表1は，数式要素に対応したキー文字／文字列の一部の実施例を示しており，これに限定されると解釈されるべきではない．表1より容易に判るように，本表記法で定めた数式要素に対応するキー文字／文字列はASCIIコードからなり，数式要素を連想しやすい頭文字やTeX,LaTeXなどに準じた文字列を採用してもよい．

表 1: 「数式要素キー文字／文字列」の例

数式要素	キー文字／文字列
a	"a"
$\mathbf{a}$	"a"
α	"a" または "alpha"
b	"b"
β	"b" または "beta"
$=$	"="
$+$	"+"
分数記号 $\frac{\square_1}{\square_2}$	"/"
$\int$	"i@" または "int"

表 2: 代表的な演算子構造に応じた数式要素キーを並べる順番
ただし，$\square_1, \square_2, \square_3$ はオペランドを表す．

構造の異なる演算子タイプ	演算子例	順番 1	2	3	4
暗黙積演算子	$\square_1\square_2$	$\square_1$	$\square_2$		
内挿二項演算子	$\square_1 + \square_2$	$\square_1$	$+$	$\square_2$	
前置単項演算子	$\angle\square_1$	$<$	$\square_1$		
上置単項演算子	$\widetilde{\square}_1$	$\sim$	$\square_1$		
下置単項演算子	$\underline{\square}_1$	$\_$	$\square_1$		
後置単項演算子	$\square_1'$	$\square_1$	$'$		
括弧	$(\square_1)$	$($	$\square_1$	$)$	
上付添字	$\square_1^{\square_2}$	$\square_1$	$\square_2$		
下付添字	$\square_{1\square_2}$	$\square_1$	$\square_2$		
前置上付添字	${}^{\square_1}\square_2$	$\square_1$	$\square_2$		
前置下付添字	${}_{\square_1}\square_2$	$\square_1$	$\square_2$		
下付上付添字	$\square_{1\square_2}^{\square_3}$	$\square_1$	$\square_2$	$\square_3$	
分数	$\frac{\square_1}{\square_2}$	$\square_1$	$/$	$\square_2$	
二乗根	$\sqrt{\square_1}$	root	$\square_1$		
n 乗根	${}^{\square_1}\sqrt{\square_2}$	$\square_1$	root	$\square_2$	
積分単項演算子	$\int\square_1$	int	$\square_1$		
積分二項演算子	$\int_{\square_1}\square_2$	int	$\square_1$	$\square_2$	
積分三項演算子	$\int_{\square_1}^{\square_2}\square_3$	int	$\square_1$	$\square_2$	$\square_3$
縮約三項演算子	$\sum_{\square_1}^{\square_2}\square_3$	sum	$\square_1$	$\square_2$	$\square_3$
内挿三項演算子	$\square_1 \overset{\square_2}{\Rightarrow} \square_3$	$\square_1$	$->$	$\square_2$	$\square_3$
前置上付演算子	$\sin^{\square_1}\square_2$	sin	$\square_1$	$\square_2$	
前置下付演算子	$\log_{\square_1}\square_2$	log	$\square_1$	$\square_2$	
lim 演算子	$\lim_{\square_1}\square_2$	lim	$\square_1$	$\square_2$	

例えば，変数aやαなどはいずれもキー"a"で表す．ただし，αに対応するキーは"alpha"も許される．これ以降，本論文では，キー文字／文字列をダブルクォーテーション"で明示的に挟んで表す．また，演算子の場合は＋記号や積分記号のように同じ記号であっても単項演算子や二項演算子など，構造の異なるものが存在する．

表2は，数式の構造を決定している代表的演算子タイプおよびその演算子構造に対応する本表記法のためのキーを並べる順番を定めた実施例を表している．例えば，積分記号はキー"i@"または"int"などで表すが，表2のように3種類の構造に対応し得る．ここで，□は一つのオペランドを表す．表2より容易に判るように，所望する数式を構成している数式要素のキーを線形に並べる順番として，ユーザの負担を少なくするために，人がその数式を読む（認知する[13],[14]）自然な順番を採用してもよい．

例えば，2次元形式の数式$\alpha=2$，$2a\beta$，$\frac{1}{a^2+1}$および$\int_a^b c=2$の本表記法（1），(2）に従った線形文字列形式は，それぞれ

a=2	(aは2に等しい)	(A)
2ab	(2掛けるa掛けるbeta)	(B)
1/a2+1	(aの二乗足す1分の1)	(C)
intabc=2	(aからbまでのcの積分は2に等しい)	(D)

となる．特に，他の表記法と大きく異なる点は，暗黙積や冪乗演算子のように表記されない記号は，線形文字列に含めないところである．例えば，式(C)に含まれるa^2は，"a2"と表記する．

3.2 新数式入力方式の概要

このように，本発明で使用する線形文字列形式表記法は，単純・簡潔になっている．その代わり，所望する数式を構成している数式記号のスタイルや要素間の区切りや各演算子に対するオペランドの範囲などが省略されており，入力された線形文字列形式の情報だけでは2次元形式が一意的に定まらず，完全にフォーマットすることはできない．そのため，本システムは線形文字列形式を解釈して構成要素を代表するキーの列に分解し，2次元形式の候補を算出するために，表3のようなキー辞書データを保持しており，各キーに関連付けられたさまざまな数式要素が登録された候補領域（対応する要素の集合）からデータの優先順位に基づいて候補を算出できるようになっている．そこで，本システムはそのキー辞書データを用いて，数式構成要素ごとに不足情報を補った候補を提示し，ユーザに候補を選択するための簡単な指示を要求する．そのような数式の構築過程を図1に示す．ユーザは数式構成要素ごとに判断すればよいので，複雑な式であっても迷うことはない．すべての構成要素が確定すれば2次元形式の構築が完了する．

3.3 数式構築アルゴリズムの考え方

ここで，システムが線形文字列形式からいかにして2次元形式を構築し得るかについて解説する．

表 3: 「数式要素キー辞書データ」の例

数式要素キー	数式要素の候補領域	優先順位
"2"	2	1
"a"	a, **a**, a, α	4, 2, 1, 3
"b"	b, **b**, b, β	4, 2, 1, 3
"alpha"	α	1
"beta"	β	1
"="	$\square_1 = \square_2$, $\square_1 \doteq \square_2$, $\square_1 \equiv \square_2$, $\square_1 \approx \square_2$, $\square_1 \cong \square_2$	1, 2, 3, 4, 5
"+"	$\square_1 + \square_2$, $+\square_1$, $\square_1 \pm \square_2$, $\pm\square_1$, $\square_1 \oplus \square_2$, $\uplus\, \square_1$	1, 2, 3, 4, 5, 6
"/"	$\square_1/\square_2$, $\frac{\square_1}{\square_2}$, $\square_1 \oslash \square_2$	1, 2, 3
空キー (省略演算子)	$\square_1\square_2$, $\square_1^{\square_2}$, $\square_{1\square_2}$, ${}^{\square_1}\square_2$, ${}_{\square_1}\square_2$	1, 2, 3, 4, 5
"i@" または "int"	$\int \square_1$, $\int_{\square_1} \square_2$, $\int_{\square_1}^{\square_2} \square_3$	1, 2, 3

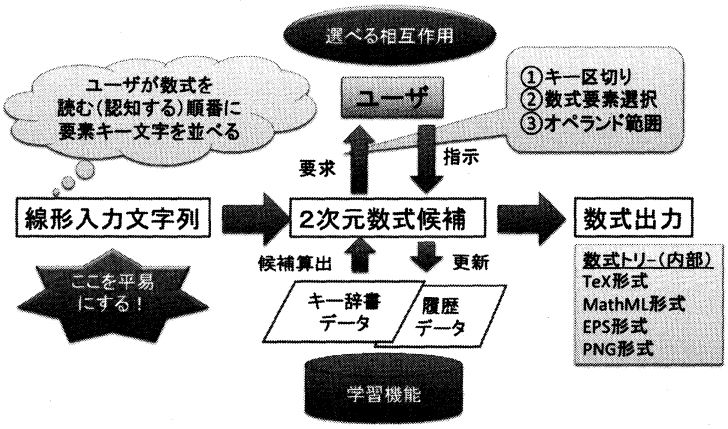

図 1: 新方式の数式構築過程

上述のように，2次元形式の数式は，数式最小単位記号自身でなければ，1個の演算子とその演算子が作用する1個または複数のオペランドによって構成される．すなわち，一般の2次元形式の数式は数式最小単位記号自身であるか，演算子を上位としてそのオペランドを下位にもつような階層構造をもち，オペランドもまた同様の階層構造をもった数式となっている．

このことから，階層構造をもつ場合の線形文字列形式は，その最上位にあたる1個の演算子を代表するキーとオペランドに相当する線形文字列の部分が表2の演算子タイプに応じた順番で並んでいると解釈できる．すなわち，この解釈が数式の階層構造を把握し，2次元形式を構築することに他ならない．本論文では，線形文字列形式に対する上記の解釈を「構造解釈」と呼ぶ．このとき，オペランドに相当する線形文字列の部分はまた独立した線形文字列形式を成し，同様に構造解釈される．

例えば，式 (C) は分数演算子キー"/"に対して，2個の線形文字列"1"および"a2+1"がオペランドに対応していると解釈できる．ただし，線形文字列 (C) だけでは，キー"/"

が最上位の演算子であるかどうかは判断できない.

したがって，一般に，本表記法（1），(2）で定めた線形文字列形式はユーザが所望する2次元形式に対して，次のような情報が不足している.

不足情報1) 各構成要素キーの分離点はどこか？

不足情報2) 各構成要素キーの所望する構成要素は候補領域のどの要素か？

不足情報3) もし確定した要素が演算子の場合，そのオペランドは線形文字列のどの部分に相当するか？

上記の不足情報は，自動的に決定することはできないため，本システムは，与えられた線形文字列に含まれる各キー文字／文字列に対して，ユーザにその不足情報を要求し，その指示を受諾することによって確定する.

不足情報2)の場合，キー辞書データ（表3）における対象キーの候補領域の中からユーザに所望する要素を選択させる方法は，例えば，かな漢字変換のように，然るべき優先順位に基づいて要素候補をキーボードに割り当てられた指示キーを押すことによって1個ずつ切り替えて提示し，所望する要素が表示されたときに採択させる方法と候補領域のすべての要素を同時に提示し，ポインティングデバイスなどを用いて選択させる方法のいずれかまたはその両方によって実施する.

また，不足情報3)のオペランドに相当する線形文字列の部分（オペランド範囲）を確定する方法は，次に示すオペランド範囲の終点候補をユーザに選択させることによって実施する．ある演算子に対するオペランド範囲の開始点となるキーは，与えられた線形文字列形式の内，必ず(a)先端キー，(b)演算子直後のキー，(c)直前オペランドの終点直後のキーのいずれかとして自動的に決まる．一方，オペランド範囲の終点となり得るキーは，一意的には決まらず，開始点以降の(d)区切りポイント直前のキー，(e)演算子直前のキー，(f)終端キーのいずれかである．システムは，確定した演算子タイプから表2のようなオペランドの個数と演算子およびオペランドが並ぶ順番の情報を用いて，与えられた線形文字列形式を構造解釈し，終点候補となるキー(d)〜(f)を機械的に算出できる．終点候補（(d)〜(f)）は線形に並んでいるため，これをユーザに選択させることは容易であろう．例えば，線形文字列(C)の演算子キー"/"が分数の場合に，分母にあたるオペランド範囲の終点は"a"，"2"，"b"の3通りあり，それぞれ(d)，(e)，(f)の場合に該当する.

以上により，すべての構成要素キーが確定したとき数式の構築が完了する.

3.4 キー辞書と学習機能

提案方式の利点の一つは，ユーザが入力した線形文字列形式から数式構成要素を解釈して2次元形式の数式を構築するための数式要素キー辞書データに自由度があり，学習機能を持たせることを可能にした点である．そのシステムの仕組みについて解説する.

当該システムは，数式要素キー辞書データをコンピュータ可読な記録データとしてユーザの利用頻度に応じて更新し，管理するデータ管理モジュールを含み得る．キー辞

書データには，表3のような文字キーなどの固定キーと利用頻度の高い式を優先的に候補とするための履歴キー文字列を含み得る．

データ管理モジュールは，入力処理が開始される前に，キー辞書データをそのコンピュータ可読な記録データが保存されている補助記憶装置からメモリに読み込むことができる．

データ管理モジュールは，対象数式要素がユーザ指示により確定されると，その（候補を算出するための）優先順位を上位になるように更新することができる．例えば，数式$\alpha=2$の入力が完了した直後には，キー"a"に対する記号αの優先順位が上位に更新され，次に本システムで構築しようとする数式では，キー"a"に対して，記号αが候補として上位に挙がるため，同じ記号を使用した式を繰り返し構築する場合に効率が上がる．

データ管理モジュールは，構築が完了した2次元形式の数式に含まれる，有効と判断された数式およびその部分とそのそれぞれのキー文字列を全て算出し，それらを新たな数式要素の候補として，現在の履歴キー辞書データに含まれる場合は優先順位を更新し，含まれない場合は新たにそのキーレコードを追加することができる．

データ管理モジュールは，履歴キー辞書データのキーレコード数を用途に合わせて有限とし，もし，追加レコードが有限レコード数を超過した場合は優先順位の最も低いレコードから破棄することができる．

データ管理モジュールは，更新されたキー辞書データのうち，長期記憶が必要と判断されたレコードを，(次回の数式構築処理に反映されるように,）補助記憶装置を用いて，コンピュータ可読な記録データとして記録することができる．

例えば，数式$\frac{1}{a^2+1}$の入力が完了した直後には，履歴キー文字列"1/a2+1"およびその部分文字列"a2+1"に対するそれぞれの数式$\frac{1}{a^2+1}$およびa^2+1が履歴キー辞書データに記録／更新される．

このように，データ管理モジュールによって，ユーザの利用頻度の高い候補を優先的に算出する事ができるようになり，ユーザが線形文字列形式から2次元形式の数式を少ないステップで構築することができるため，さらに効率が上がる．

4　数式入出力システムの実施例

本章では，望ましい実施形態をいくつかの実施例を基に解説する．

4.1　擬似コードによる実施例1

表4に示した擬似コードは，数式$\alpha=2$を構築する過程を，ステップごとの実施画面とユーザの指示コードで表したものである．

まず，ステップ1で線形文字列形式(A)をキーボードを用いて入力し，構築処理の開始指示（例えば Enter キー）を打つ．システムは線形文字列形式を解釈して2次元形式の候補を算出し，ステップ2のように表示する．ボックスで強調された部分は，選択対象の構成要素を表しており，所望の記号でないため，所望の記号が現れるまで次候補要求指示（例えば Space キー押下）を繰り返す（ステップ3）．実際，システムのキー辞

書データ（表3の例で）は，キー"a"に関連する候補として4個の表記コード（ローマン体，ボールド体，イタリック体，ギリシャ文字）が登録されており，優先順位に基づいて現在の候補要素のみが提示されるようになっている．ステップ4で所望する表記コードが表示されたので，対象要素採択指示 Enter を打ち，次の構成要素に対象を移す．ステップ5では選択対象が"="演算子であるため，2個のオペランドに，範囲をあらわすアンダーラインがマークされている．また，両者の内，第2オペランドの範囲が二重線になっているのは，現在の範囲変更対象であることをユーザに知らせている．オペランド範囲の変更については次の事例で紹介する．表3のキー辞書データの例では，"="演算子キーに関連して5個の演算子が登録されている．各演算子は，その演算子記号表記コードに加えて，演算子の作用構造を区別する表2のいずれかに対応した演算子タイプの情報も保持している．対象要素採択指示 Enter により最後の構成要素が確定すれば，数式の構築が完了する（ステップ6）．

ところで，本線形文字列形式表記法では，キー"a"の代わりに"alpha"のような冗長な表現を許し得る．例えば，表4のステップ1において，式(A)と同じ意味を表す線形文字列"alpha = 2"を入力し構築処理を開始すると，いきなりステップ4に進むことができる．このように，冗長な表現は候補を絞ることができるため数式を構築し易くなる．

4.2 擬似コードによる実施例2

表5に示した擬似コードは，数式 $\frac{1}{a^2+1}$ を構築する過程を，ステップごとの実施画面とユーザの指示コードで表したものである．

まず，ステップ1で，線形文字列形式(C)を入力し，構築処理の開始指示 Enter を打つ．ステップ2では，演算子キー"/"に対する候補とオペランド範囲がマークされている．次候補要求指示 Space を押して，所望する分数記号を表示させる．ステップ3で，対象要素採択指示 Enter を押し，次の構成要素キー"a"に対象を移す．ステップ4では，変数 a が候補として強調表示されている．そのまま対象要素採択指示 Enter を押し，次の構成要素に対象を移す．このとき，数式最小単位記号が並ぶ"a"と"2"の間には，省略演算子があると解釈され，ステップ5では，その省略演算子の第1候補（表3の空キー参照）である暗黙積演算子が選択対象となる．このように，本システムでは数式最小単位記号が並ぶ間を「区切りポイント」と呼んで，特別な処理がなされる．省略演算子は表示されないため対象要素を強調するためのボックスは見えないが，オペランド範囲はマークされている．このとき範囲変更対象オペランドは"2+1"となっている．そこで対象の暗黙演算子に対して次候補要求指示 Space を打つと，冪乗演算子に切り替わる（ステップ6）．しかし，指数部のオペランド範囲が所望する範囲と異なるので，範囲縮小指示（例えば ← キー，拡大する場合は → キー）を押して，ステップ7の表示画面のように範囲を変更する．そこで，冪乗演算子（ステップ7）および次の＋演算子（ステップ8）に対して，対象要素採択指示 Enter を押せば，数式の構築が完了する（ステップ9）．

4.3 擬似コードによる実施例3

表6に示した擬似コードは，数式 $\int_a^b c = 2$ を構築する過程を，ステップごとの実施画面とユーザの指示コードで表したものである．

まず，ステップ1で線形文字列形式(D)を入力し，構築処理の開始指示 Enter により2次元形式候補が算出され，提示される．このとき，システムのキー辞書データには，与えられた線形文字列をキーの列に分解する際に，例えば文字数の大きいキー文字列を優先的に検索させるような，検索優先順位の情報を含み得る．したがって，線形文字列(D)に含まれる文字列"int"は積分演算子キーと解釈される．また，文字列"abc"の部分は3個の数式最小単位記号が並んでいると解釈され，2箇所の区切りポイントが含まれる．ステップ2で，選択対象の積分演算子は初め単項演算子となっている．ここで次候補要求指示 Space を打つと，積分二項演算子に変わり，最初の区切りポイントで2個のオペランドが分離されている（ステップ3）．さらに次候補要求指示 Space を打つと，積分三項演算子に変わる（ステップ4）．しかし，第3オペランドの範囲は変更する必要があるが，範囲変更対象は第1オペランドのままである．そこで，範囲変更対象次移動指示（例えば ↑ キー，前移動指示は ↓ キー）を押して，範囲変更対象を第2オペランドへ移し（ステップ5），さらに範囲変更対象次移動指示 ↑ を押して，範囲変更対象を第3オペランドへ移す（ステップ6）．そこで範囲縮小指示 ← を押して範囲を変更する（ステップ7）．この時点で，積分演算子を対象要素採択指示 Enter により確定する．次に各オペランドの要素も順に対象要素採択指示 Enter により確定していく（ステップ8〜11）．最後に"="演算子を対象要素採択指示 Enter により確定すれば，数式の構築が完了する（ステップ12）．

5 まとめ

以上のように，本方式は，ユーザが，2次元形式の数式を読む順番，すなわち，2次元形式の数式を構成する文字列をユーザが認知する順番で，当該文字を入力すれば，入力された線形文字列が，2次元形式の数式に変換され得る構成を採用している．そのため，ユーザは，2次元形式の数式を入力するために，固有の文法を入力する必要がなくなり，ユーザの作業負担が著しく軽減されることになる．

[進歩点（独創性・優位性）]

このように，本提案方式の独創性は，数式に対する文字の入力の順番をユーザ側の認知行動に対応させた点にある．したがって，従来技術に比べ，(1) 数式構築のための入力文字列が容易，(2) 数式構築のための数式候補の選択が可能，(3) 入力頻度に対応する数式学習機能により入力効率が向上するといった優位性がある．

[想定される利用分野・用途]

産業上の応用可能性として，例えば，文書・Webページ・LMSのための数式エディタや，数式処理システムなどのフロントエンド，数式をキーワードとする情報検索ツール，数学ソフトウェアなどの数式入力補助ツールなどに組み込んで利用することが考えられる．

表 4: 実施例 1

ステップ	表示画面	指示コード
1:	a=2	Enter
2:	$\boxed{a}=2$	Space
3:	$\boxed{\mathrm{a}}=2$	Space
4:	$\boxed{\alpha}=2$	Enter
5:	$\alpha=2$	Enter
6:	$\alpha=2$	

表 5: 実施例 2

ステップ	表示画面	指示コード
1:	1/a2+1	Enter
2:	$1\boxed{/}a2+1$	Space
3:	$\frac{1}{a2+1}$	Enter
4:	$\frac{1}{\boxed{a}2+1}$	Enter
5:	$\frac{1}{a2+1}$	Space
6:	$\frac{1}{a2+1}$	←
7:	$\frac{1}{a^2+1}$	Enter
8:	$\frac{1}{a^2\boxed{+}1}$	Enter
9:	$\frac{1}{a^2+1}$	

表 6: 実施例 3

ステップ	表示画面	指示コード
1:	intabc=2	Enter
2:	$\boxed{\int}abc=2$	Space
3:	$\int_{\boxed{a}}^{b}bc=2$	Space
4:	$\int_{\boxed{a}}^{b}c=2$	↑
5:	$\int_{\boxed{a}}^{b}c=2$	↑
6:	$\int_{\boxed{a}}^{b}c=2$	←
7:	$\int_{a}^{b}c=2$	Enter
8:	$\int_{\boxed{a}}^{b}c=2$	Enter
9:	$\int_{a}^{\boxed{b}}c=2$	Enter
10:	$\int_{a}^{b}\boxed{c}=2$	Enter
11:	$\int_{a}^{b}c=2$	Enter
12:	$\int_{a}^{b}c=2$	

[今後の発展可能性]

本研究は，まだ，始まったばかりで，もし，本方式が普及すれば，数式を含む電子情報の取り扱いが容易になり，数理科学・技術分野および理数系教育の情報化の推進に効果がある可能性をもっている．

しかし，そのためにも，多くの研究課題が残されている．第 1 に，さまざまな分野の数学ソフトウェアに組み込んだ実装システムの開発を行い，実証実験を進めていく必要がある．第 2 に，ユーザビリティのさらなる向上の研究．第 3 に，数式入力の一貫性へ向けてのプロジェクト化．第 4 に，理数系教育への応用である．特に，理数系デジタル教材による生徒・学生の個別学習に有効となるような取り組みを進めていきたい．

参考文献

[1] 文部科学省: 教育の情報化ビジョン,
http://www.mext.go.jp/b_menu/houdou/23/04/1305484.htm

[2] 中村泰之:STACK と Moodle による数学 e ラーニング, 数理解析研究所講究録 *1735* 「数式処理と教育」,2011,9–15.

[3] 白井詩沙香, 福井哲夫:数式処理を用いた教育を想定したタイピング能力の調査, 数理解析研究所講究録 *1735* 「数式処理と教育」,2011,73–84.

[4] 吉長裕司, 川畑洋昭:情報教育におけるキーボードリテラシーの一考察, 情報処理学会論文誌, Vol.42 No.9, 2001,2359-2367.

[5] Donald Ervin Knuth: The TeXbook ,Addison-Wesley, 1984.

[6] http://www.w3.org/TR/2001/REC-MathML2-20010221/

[7] Microsoft co.: Microsoft(R) 数式エディターユーザヘルプ, http://office.microsoft.com/ja-jp/word/.

[8] Design Science, INc.: MathType,http://www.dessci.com/en/products/mathtype/.

[9] M.Fujimoto, T.Kanahori and M.Suzuki: Infty Editor ? A Mathematics Typesetting Tool with a Handwriting Interface and a Graphical Front-End to OpenXM Servers, Computer Algebra - Algorithms, Implementations and Applications, RIMS Kokyuroku Vol.1335 , 2003, 217-226.

[10] 藤本光史:数式文書編集ソフトウェア InftyEditor, http://www.sciaccess.net/jp/InftyEditor/index.html, サクセスネット,2002.

[11] Microsoft co.,http://www.microsoft.com/windows/windows-7/default.aspx

[12] マイクロソフトコーポレーション:数式の構築を自動化するためのシステム, 特許公開 2006-85673,2006.

[13] 高野陽太郎:認知心理学 2 記憶, 東京大学出版会,1995.

[14] 村田暑生:認知科学　心の働きをさぐる, 朝倉書店,2007.